An Introduction to Software Visualization

Dr. Jonathan I. Maletic
Software DevelopMent Laboratory <SDML>
Department of Computer Science
Kent State University

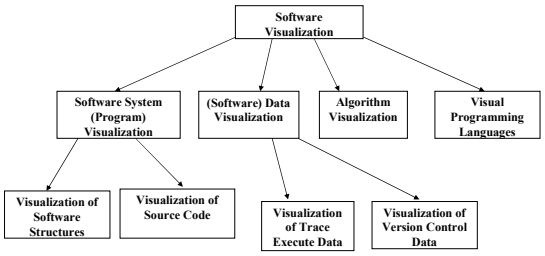
Course Overview

- Introductory Lectures
  - Software visualization – terminology, dimensions and perspectives
  - Information visualization – an overview
  - Program analysis – the input to the problem
  - Program understanding/comprehension – the model
- Papers: Information Visualization
  - 20-25 papers
- Papers: Software Visualization
  - 20-25 papers

What is Software Visualization?

"Software visualization is the use of the crafts of typography, graphic design, animation and cinematography with modern human-computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software." [Price '93, '98].

Types of Software Visualization



Visualizing Software

Software System Visualization

Software Data Visualization

Visual Programming

Algorithm Animation

Exclude

Applied to Large-scale Software

We focus on visualization environments, techniques, and metaphors that support:

- Maintenance, re-engineering, reverse engineering
- Software development
- Project management
- Understanding

of large scale software systems

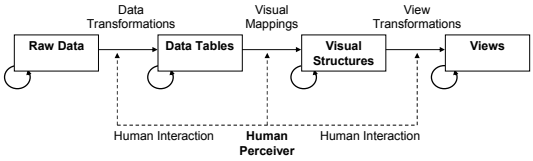
Specifics of the Problem Domain

- Visualize design and architectural information
- Reduces, in part, to (large) connected graphs
- Nodes represent complex entities i.e., software module, class, function, component, subsystem, etc.
- Edges represent abstract relationships between the nodes i.e., aggregation, association, inheritance, invocation, etc.

Taxonomies for Software Visualization

- Existing taxonomies (Price, Roman, Myers, Stasko) are very broad and detailed
- Need to emphasize software engineering tasks involved in building and maintaining large-scale software systems
- No single software visualization tool can address all SE tasks

Reference Model for Visualization



Raw Data: idiosyncratic formats
Data Tables: relations (cases by variables) + meta data
Visual Structures: spatial substrates + marks + graphical properties
Views: graphical parameters (position, scaling, clipping, etc.)

Visualization can be described as a mapping of data to visual form that supports human interaction for making visual sense [Card '99].

Task Oriented View for Maintenance and Development of Large Systems

Task – *why* is the visualization needed?
Audience – *who* uses the visualization?
Target – *what* to represent?
Representation – *how* to represent?
Medium – *where* to represent?

Task

- Support for large scale, industrial-size, software systems and processes.
- Supports the understanding/comprehension (cognitive) process
- This is the driving force behind classification of software visualization systems (given our perspective)

Specific Tasks

- Development:
  - Design, Product evolution
  - Programming
  - Testing, Debugging
- Maintenance:
  - Fault detection
  - Reverse engineering, Re-engineering
  - Impact analysis
- Management
  - Version control
  - Resource allocation

Audience

- Experienced developers can handle multiple abstraction levels
  - they need access to both design- and code-level information, as well as to the dynamic features.
- Project managers
  - they need access to design- and process-level information.
  - they might not be skilled programmers.

Target

- Static features (relationships)
  - Design and architectural level information
  - Source code level information
  - Documentation
- Dynamic features (behavior)
  - Control and data flow at execution
  - Trace information

Representation

- User centric (versus compiler centric) – the visualization should present features of the software in concepts from the user's universe.
- Cognitive based - the building blocks of the visual language must map to natural concepts and abstractions
- Does not overload the user – each element should have multiple attributes, but there should be a limit on the diversity. This limit should be driven by cognitive factors and the medium.

Additional Features of Representation

- Support multiple levels of abstraction:
  - Source code
  - Design
  - Design Patterns
  - Architecture
- Mapping between abstraction levels (e.g., drill down)
- Support navigation within the visualization

Mediums for Software Visualization

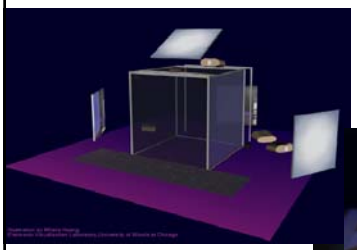
- Paper Documents
  - 2D, poor navigation, static
- White board
  - 2D, static
- The Desktop Display
  - High resolution but limited display area
  - 2D+ (for the most part)

New Types of Software Visualization Mediums

- More real estate
  - Multiple displays, large displays
- Support for collaborative problem solving
- Utilize:
  - 3D, 3D+
  - Virtual Reality (combine stereopsis with motion), Immersive Environments
  - Stereo displays, Multi-resolution displays
  - Multi-type medium (e.g., laptop + VE)
- Provide navigational controls

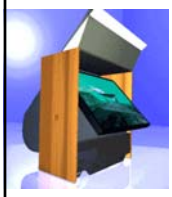
The CAVE

Commercialized by
Pyramid Systems
&
VRCO



Images from EVL

The ImmersaDesk I, II, III

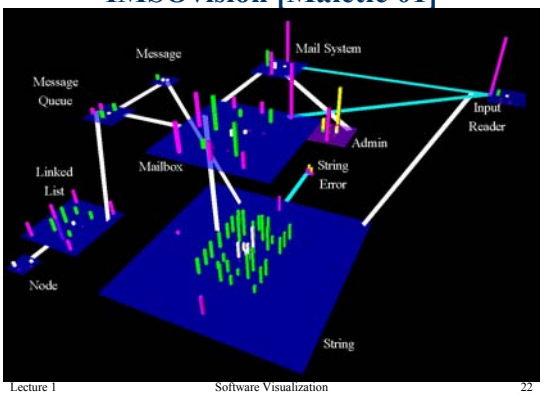


Images from EVL

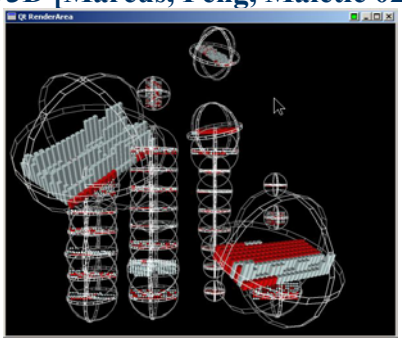
Research Directions

- Focus on SE Task(s)
- What types of (representations, mediums, etc) best support particular tasks?
- Investigate new mediums and representations
- Environments that support collaborative development
- Utilize existing research in cognitive psychology

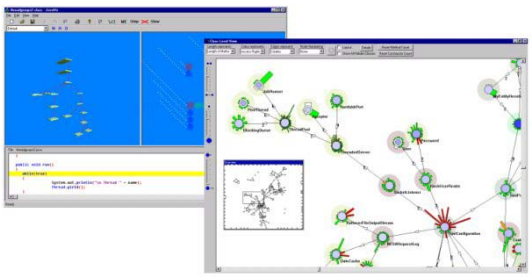
IMSOvision [Maletic 01]



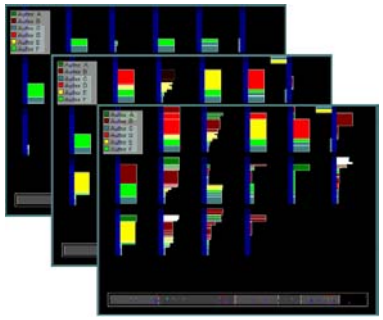
sv3D [Marcus, Feng, Maletic 02]



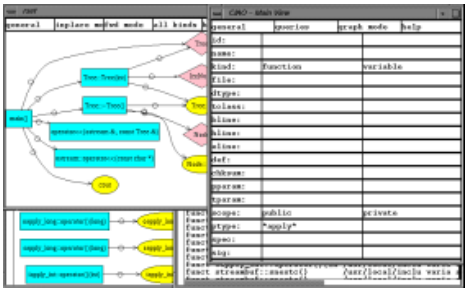
DJVis [Smith & Munro 02]



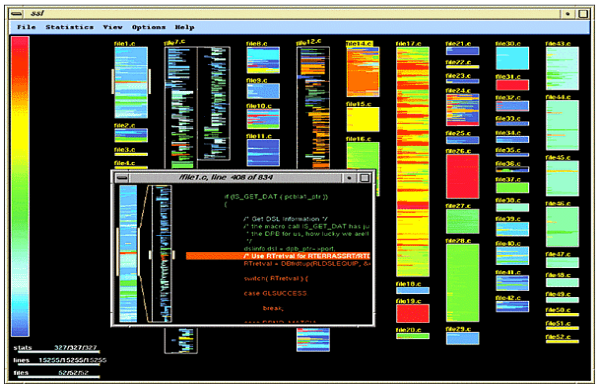
Revision Towers [Taylor & Munro 02]



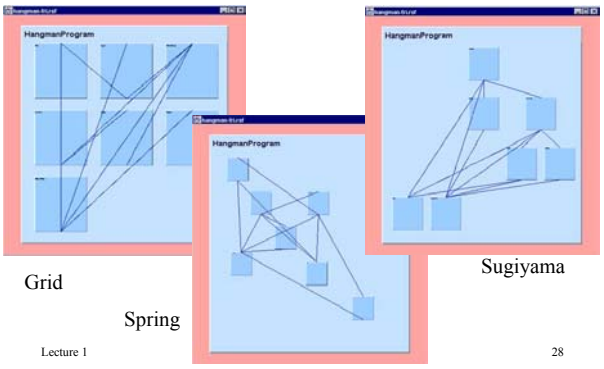
Acacia: CIAO for C++ [AT&T Research]



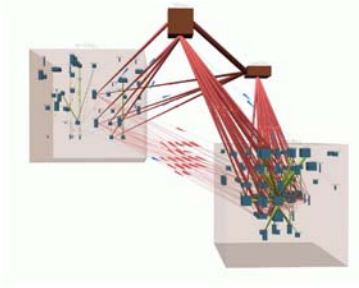
SeeSoft -Code Version History [Eick96]



SHriMP [Storey01]



NV3D – [Ware ’97]



Geon – [Ware ’00]